A

Project Report

On

DESKTOP AI

Submitted in the partial fulfilment of the requirement for the degree of

BACHELOR OF COMPUTER APPLICATIONS

(Session: 2024-2025)

GOVERNMENT COLLEGE HANSI

Under the Supervision of:

Mr.Anil Kumar

(Assistant Professor of Computer Science)

Submitted by:

Name: Sonu

Roll No:223122220010

Class: B.C.A.(6th sem)

DEPARTMENT OF COMPUTER SCIENCE

GOVERNMENT COLLEGE, HANSI

Affiliated to

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE & TECHNOLOGY, HISAR

Project Report

On

DESKTOP AI

GURU JAMBHESHWAR UNIVERSITY OF SCIENCE & TECHNOLOGY, HISAR

In the partial fulfilment of the requirement for the degree of

BACHELOR OF COMPUTER APPLICATIONS

(Session: 2024-2025)

Under the Supervision of:

Mr. Anil Kumar

(Assistant Professor of Computer Science)

Submitted by:

Name: Sonu

Roll No:223122220010

Class: B.C.A.(6th sem)

GOVERNMENT COLLEGE,HANSI

CERTIFICATE

This is to certified that Miss **Sonu**,Roll Number: **223122220010** a bonafide student of Bachelor of Computer Applications program being run by **GOVERNMENT COLLEGE , HANSI** of batch 2022-2025 has complete the project entitled **"DESKTOP AI"** under my supervision and my guidance. It is further certified that the work done in this project is a result of candidate's own efforts. I wish her all success in her life.

Dr. Banta Singh Jangra

(Head of department computer science)

Supervisor:

Mr.Anil Kumar

Assistant Professor
Govt. college ,Hansi

DECLARATIONS

I hereby certify that the work which is being presented in the Project Report entitled "**DESKTOP AI"** by "**Sonu**" in partial fulfillment of requirement for the award of degree of BCA submitted in the department of computer science and applications, GOVERNMENT COLLEGE, HANSI under **"GURU JAMBHESHWAR UNIVERSITY OF SCIENCE & TECHNOLOGY,Hisar"** is an authentic record of my own work carried out during a period of 2022-2025 under the supervision of Mr. Anil Kumar the matter presented in this project has not been submitted in any other university/institute for the award of BCA degree.

Sonu

This is to certified that the above statement made by the candidate is correct to best of our knowledge.

Under Supervision of:

Mr. Anil Kumar

Assistant Professor

Govt. College ,Hansi

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose corporation made it possible, whose constant guidance and encouragement crown all efforts with success. Every possible effort is dedicated to **Dr. Banta Singh Jangra** and my internal supervisor **Mr. Anil Kumar(Assistant Professor of Computer Science),** for giving significance to our endowment regarding this project. I am highly indebted for the gesture, invaluable suggestion and boosting confidence to make this successful.

I am thankful to my entire respected faculty members who were a great source of information and knowledge for me. For the guidance, inspiration and constructive suggestion that helped me in the preparation of this project.

I dedicated my whole effort in this project to my parents and friends, who assisted me with all kind of moral as well as monetary support throughout my project work. I also extend my apologies for all errors or omissions. Which is solely my responsibility.

Sonu

INDEX

1. ABSTRACT

"DESKTOP AI" is an advanced, AI-powered project designed to assist users with a wide range of tasks through natural language processing. This versatile system integrates various Python libraries to deliver functionalities such as voice recognition, text-to-speech synthesis, web browsing, and task management. By incorporating technologies like speech recognition, web scraping, multimedia handling, and real-time internet speed testing, DESKTOP AI aims to streamline productivity and provide users with a seamless, interactive experience. With its intuitive interface and multi-functional capabilities, it offers an efficient, hands-free solution for both personal and professional use.

2. INTRODUCTION

The Kanha Virtual Assistant is designed to facilitate everyday tasks through voice commands, aiming to streamline user interactions with their computing environment. It is built using Python, with integration of multiple libraries and technologies to ensure robust performance. The assistant can recognize user commands, process them, and provide relevant responses or perform actions accordingly.

Features of the project

1.Voice Recognition & Response: Utilizes speech recognition to convert speech to text. Uses pyttsx3 for text-to-speech conversion to provide vocal responses.

2. Task Management: Can schedule and show daily tasks. Capable of clearing old tasks and adding new ones to a text file.

3.Multimedia Control: Plays music from a specified directory. Controls system volume through keyboard automation. Plays videos and controls playback (pause, play, mute).

4.Web Interaction: Searches for information on Google and Wikipedia. Searches for videos on YouTube and opens specific URLs. Generates and saves QR codes.

5.System and Network Utilities: Checks and reports internet speed. Displays the current location based on IP address and shut down the system upon command.

6.Entertainment: Plays a game of Rock-Paper-Scissors. Sends a pre-defined WhatsApp message.

7.Text Translation: Translates input text to Hindi using googletrans and gtts for voice playback.

8.Notifications: Sends desktop notifications for schedule reminders and other updates

3. Technical Implementation

Libraries and Modules:

speech_recognition: For recognizing user speech.

pywhatkit: For searching on YouTube and WhatsApp messages.

googletrans: For translating text.

requests and BeautifulSoup: For web and fetching news.

qrcode: For generating QR codes.

plyer: For desktop notifications.

pygame: For playing sound files.

pyautogui: For automating GUI interactions.

pyttsx3: For converting text to speech.

System Architecture:

Voice Command Processing: Receives and interprets user commands through microphone input.

Action Execution: Executes commands based on predefined functions such as playing music, opening applications, or fetching data from the web.

Feedback & Notifications: After executing the action, the assistant provides real-time feedback through vocal responses using text-to-speech synthesis, on-screen notifications, and terminal output for status updates.

Error Handling & Recovery: The assistant includes robust error-handling mechanisms, detecting invalid commands, managing system errors gracefully, and offering retry options or alternative suggestions.

Command Classification & Routing: Once processed, the commands are categorized into specific action types such as media control, information retrieval, system operations, or task management, and then routed to the corresponding function module.

Action Execution Layer: The assistant maps the interpreted commands to predefined functions, executing actions such as media playback, web interaction, system control, and real-time information retrieval.

4. User Interaction Flow

Initialization:

The user must enter a password and say a specific phrase to activate the assistant.

Voice Commands:

The assistant listens for commands and processes them using speech recognition. Commands are categorized and directed to the appropriate function (e.g., "play music", "search YouTube", "check internet speed").

Response and Actions:

Based on the command, the assistant provides vocal responses or performs actions (e.g., playing a song, generating a QR code). Feedback is given through speech and notifications as needed.

Error Handling:

The assistant includes error handling for invalid commands or failed operations, ensuring a smooth user experience.

Graceful Exit:

The assistant offers a **smooth shutdown** process when the user says commands like ″go to sleep″ or ″finally sleep″, ensuring all processes terminate properly.

Multi-tasking Capabilities:

The assistant can handle multiple operations in succession, allowing users to give continuous commands without needing to re-activate it.

Visual Feedback:

The assistant displays relevant information in the terminal or opens web pages when requested (e.g., YouTube search results or a news website).

5.REQUIREMENT SPECIFICATION

INTRODUCTION:

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These prerequisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system application is the physical computer resources, also known as hardware. A hardware requirement list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following subsection discuss the various aspects of hardware requirements.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT: The hardware minimum and maximum recommended requirements are listed below:

Hardware	Minimum requirement	Maximum requiurement
CPU	Intel core i3	i3 beyond
RAM	4gb	8gb or more
STORAGE	500 mb	1gb free disk space
SOUND CARD	Basic sound card	High quality sound card

 Disk Drives: Each client computer must have sufficient disk space to store the client portion of the software along with any necessary local data files. It is recommended to provide a **local disk drive** for each client computer to ensure optimal performance. However, in **Client/Server environments**, the system can operate using **"diskless workstations"**, where all disk access is handled by the network file server. For the **database server**, the minimum recommended hard disk capacity is **4.1 GB**, but for better performance and future scalability, a **capacity of 8.2 GB or higher** is preferred. Regular **disk maintenance** and cleanup are advised to ensure efficient storage utilization.

 Mouse: A **functional mouse** is essential for smooth navigation and interaction with the **client software**, especially when running under **Windows OS** or similar GUI-based environments. For **precision and accuracy**, an **optical or laser mouse** is recommended. In cases where the assistant needs gesture-based controls, a **multi-touch trackpad** or **external pointing device** may also be beneficial.

 Keyboard: Each client system must be equipped with a **104-key extended keyboard** to support full functionality, including navigation keys, function keys, and a numeric keypad. For improved efficiency, a **mechanical or ergonomic keyboard** is recommended, especially for users handling frequent data input or prolonged usage.

 Processor & RAM: The client computers should have a **multi-core processor** (e.g., **Intel i5/i7** or AMD equivalent) to handle speech processing and command execution efficiently. A minimum of **8 GB of RAM** is recommended, with **16 GB or higher** preferred for smooth multitasking and optimal performance.

 Network Requirements: For **network-based operations**, a stable and fast internet connection is essential. A **wired Ethernet connection** is preferred for database servers, while clients can use **Wi-Fi** with at least **100 Mbps** speed to ensure smooth communication and minimal latency.

 Display: The client computers should have a **minimum resolution of 1366x768** pixels, although **1920x1080 (Full HD)** or higher is recommended for better visibility and user experience. For multitasking, **dual-monitor setups** are also beneficial.

 Audio Input & Output: To support **voice command processing**, the system requires a **high-quality microphone** for clear input and **speakers or headphones** for vocal responses. For improved accuracy, a **noise-canceling microphone** is recommended.

 Power Supply & Backup: To ensure uninterrupted operations, client systems and servers should be connected to a **stable power supply** with **UPS (Uninterruptible Power Supply)** backup to prevent data loss or corruption during power outages.

 Operating System Compatibility: The Kanha Virtual Assistant is compatible with **Windows 10/11**, **Linux**, and **macOS**. It is recommended to use the latest OS version with regular updates for enhanced security and compatibility.

 Software Dependencies: The system requires **Python 3.x**, along with necessary libraries and modules such as **SpeechRecognition, Pyttsx3, requests, and OpenAI APIs**. It is advisable to keep these libraries updated to ensure compatibility with the latest features.

SOFTWARE REQUIREMENTS:-

Software Requirements deals with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed

	MINIMUM REQUIREMENT	MAXIMUM REQUIREMENT
OPERATING SYSTEM	Windows 10 or newer	Windows 10 or newer
IDEAL USED	Python 3.7 or newer	Python 3.9 or newer

6.Coding

```
from openai import OpenAI                    # chatbot like interaction
import pyaudio                               # for input and output of audio
from geopy.geocoders import Nominatim # converting address in latitude
import asyncio                               # use for asynchronous sleep
import subprocess                            # use for open notepad
import webbrowser                            # use for open google
import pyttsx3                               # use for converting text to speech
import qrcode                                # use for generate qr code and saving it
import datetime                              # for current time
import speech_recognition as sr #create a recognizer (no speech recognisation permised in one line)
import requests                              # to interact with websites and apis
from bs4 import BeautifulSoup          # for html or xml document joined with request
import os                                    # use for interact with system
import psutil                                # system performance monitoring like process ending starting
import pyautogui                             #moves the mouse to a position
import wikipedia                             # searching for something "like python"
import pywhatkit                             # sending whatsapp messages instantly
import random                                #generate random number
from plyer import notification     #desktop notification displaying
from pygame import mixer          # initialize sound mixer
import speedtest                             # for internet speed
from pynput.keyboard import Key,Controller# for keyboard keys controller
from time import sleep                       # pause execution for some time
from googletrans import Translator           # translating something
from gtts import gTTS                        #convert speech and saving somewhere in file
import googletrans                           # for translating something
import geocoder                              # get ip location
from playsound import playsound   #plays generated speech
```

```
import time                              #work related with time
import json                              #writes a sample dictionary from apis
from tkinter import *                    #opens a gui window
from PIL import Image,ImageTk,ImageSequence # manipulating images
keyboard = Controller()
mixer.init()
for i in range(3):
    a = input("Enter Password to get help from kanha :- ")
    pw_file = open("password.txt","r")
    pw = pw_file.read()
    pw_file.close()
    if (a==pw):
        print("WELCOME ! PLZ SPEAK [radhe radhe] TO LOAD ME UP")
        break
    elif (i==2 and a!=pw):
        exit()
    elif (a!=pw):
        print("Try Again")
root = Tk()
root.geometry("1000x500")
# function for playing a glf in starting to give a graphical interactive interface
def play_gif():
    root.lift()
    root.attributes("-topmost",True)
    global img
    img = Image.open("nova.gif")
    lbl = Label(root)
    lbl.place(x=0,y=0)
    i=0
    mixer.music.load("Startup2.mp3")
    mixer.music.play()
    for img in ImageSequence.Iterator(img):
        img = img.resize((1000,500))
```

```
        img = ImageTk.PhotoImage(img)
        lbl.config(image=img)
        root.update()
        time.sleep(0.02)
    root.destroy()
play_gif()
root.mainloop()
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)
#function for speaking something by the engine
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
#function for taking voice command
def takeCommand():
    r = sr.Recognizer()
    r.energy_threshold = 100  # Ignores background noise/music
    r.dynamic_energy_threshold = True  # Auto-adjust to environment
    with sr.Microphone() as source:
        print("Listening your command ...")

        r.adjust_for_ambient_noise(source, duration=1)  # Noise adaptation
        try:
            audio = r.listen(source, timeout=5, phrase_time_limit=5)  # Capture voice
        except sr.WaitTimeoutError:
            print("No speech detected. Try again.")
            return "None"
    try:
        print("Processing speech...")
        query = r.recognize_google(audio, language='en-in')  # Recognize speech
        print(f"You said: {query}\n")
```

```
        return query.lower()  # Convert to lowercase for easy comparison
    except sr.UnknownValueError:
        print("Couldn't understand the command. Please repeat.")
        return "None"
    except sr.RequestError:
        print("Network error. Check your internet connection.")
        return "None"
#function for greeting in starting
def greetMe():
    hour  = int(datetime.datetime.now().hour)
    if hour>=0 and hour<=12:
        speak("radhe radhe ,goodmorning")
    elif hour >12 and hour<=18:
        speak("radhe radhe, Good Afternoon ")
    else:
        speak("radhe radhe Good Evening")
    speak("Please tell me, How can I help you ?")
#function for playing the music from existing music folder from the system system
def play_music():
    music_folder =  "C:\\Users\\mehta\\Music"
    songs = os.listdir(music_folder)
    song = random.choice(songs)
    song_path = os.path.join(music_folder, song)
    os.startfile(song_path)
#function for closing file
def close_file():
    speak("Please specify the file or application name you want to close.")
    file_name = takeCommand().strip().lower()  # Take voice input
    if not file_name or file_name in ["cancel", "exit", "stop"]:
        speak("Okay, cancelling the request.")
        return
    found = False  # Flag to check if the file is found
    # Loop through all running processes
```

```
    for process in psutil.process_iter(['pid', 'name']):
        try:
            process_name = process.info['name'].lower()
            if file_name in process_name:  # Check if filename matches
                os.system(f"taskkill /F /PID {process.info['pid']}")  # Force kill
                speak(f"Closing {process_name}")
                found = True
                return
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            continue  # Skip any processes that can't be accessedif not found:
        speak("File or application not found. Please check the name and try again.")
#function for file opening
def open_file():
    speak("Please specify the file name or full path.")
    file_name = takeCommand().strip().lower()  # Take voice input
    if not file_name or file_name in ["cancel", "exit", "stop"]:
        speak("Okay, cancelling the request.")
        return
    # If the user provides a full file path
    if os.path.isfile(file_name):  # Check if file exists
        os.startfile(file_name)
        speak(f"Opening {file_name}")
        return
    # Get the current user's home directory
    user_home = os.path.expanduser("~")  # Fetches C:\Users\YourUsername
    # Common directories to search
    common_dirs = [
        os.path.join(user_home, "Documents"),
        os.path.join(user_home, "Downloads"),
        os.path.join(user_home, "Desktop")
    ]
    found = False  # Flag to check if the file is found
    for directory in common_dirs:
```

```
        for root, _, files in os.walk(directory):  # Walk through folders
            for file in files:
                if file_name in file.lower():  # Check if filename matches
                    file_path = os.path.join(root, file)
                    os.startfile(file_path)
                    speak(f"Opening {file}")
                    found = True
                    return
    if not found:
        speak("File not found. Please try again with the correct name or full path.")
# function for search song on the youtube
def search_on_youtube(song_name):
    try:
# Construct the YouTube search URL
search_url=f"https://www.youtube.com/results?search_query={song_name.replace(' ', '+')}"
        # Open the web browser and search for the song on YouTube
        webbrowser.open(search_url)
    except Exception as e:
        print("An error occurred:", str(e))
        print("Sorry, I couldn't perform the search.")
# function for checking the internet speed
def check_internet_speed():
    st = speedtest.Speedtest()
    st.get_best_server()
    download_speed = st.download() / (1024 * 1024)  # Convert bytes to megabits
    upload_speed = st.upload() / (1024 * 1024)  # Convert bytes to megabits
    ping = st.results.ping
    print(f"Download Speed: {download_speed:.2f} Mbps")
    print(f"Upload Speed: {upload_speed:.2f} Mbps")
    print(f"Ping: {ping} ms")
    speak(f"Download Speed is {download_speed:.2f} Mbps")
    speak(f"Upload Speed is {upload_speed:.2f} Mbps")
    speak(f"Ping is {ping} milliseconds")
```

```
#function for checking current location
def get_current_location():
    try:
        # Get the current location based on IP address
        location = geocoder.ip('me')
        if location:
            return location.address
        else:
            return "Location not found."
    except Exception as e:
        print(f"Error occurred while getting location: {str(e)}")
        return "Location not found."
# for generating qr code which u want to make like for instagram , whatsapp, phonepay etc
def generate_qr_code(text_or_url, filename="qr_code.png"):
    try:
        # Create QR code instance
        qr = qrcode.QRCode(
            version=1,
            error_correction=qrcode.constants.ERROR_CORRECT_L,
            box_size=10,
            border=4,
        )
        # Add data to the QR code
        qr.add_data(text_or_url)
        qr.make(fit=True)
        # Create an image from the QR code instance
        img = qr.make_image(fill_color="black", back_color="white")
        # Save the image to a file
        img.save(filename)
        print(f"QR code generated successfully as {filename}")
    except Exception as e:
        print(f"Error occurred while generating QR code: {str(e)}")
# a small game for refreshing mind doing work
```

```
def game_play():
    print("LETS PLAYYYYYYYYYYYYYY")
    speak("Lets Play ROCK PAPER SCISSORS !!")
    i = 0
    Me_score = 0
    Com_score = 0
    while(i<5):
        choose = ("rock","paper","scissors") #Tuple
        com_choose = random.choice(choose)
        query = takeCommand().lower()
        if (query == "rock"):
            if (com_choose == "rock"):
                speak("ROCK")
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
            elif (com_choose == "paper"):
                speak("paper")
                Com_score += 1
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
            else:
                speak("Scissors")
                Me_score += 1
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
        elif (query == "paper" ):
            if (com_choose == "rock"):
                speak("ROCK")
                Me_score += 1
                print(f"Score:- ME :- {Me_score+1} : COM :- {Com_score}")
            elif (com_choose == "paper"):
                speak("paper")
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
            else:
                speak("Scissors")
                Com_score += 1
```

```
            print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
        elif (query == "scissors" or query == "scissor"):
            if (com_choose == "rock"):
                speak("ROCK")
                Com_score += 1
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
            elif (com_choose == "paper"):
                speak("paper")
                Me_score += 1
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
            else:
                speak("Scissors")
                print(f"Score:- ME :- {Me_score} : COM :- {Com_score}")
        i += 1
    print(f"FINAL SCORE :- ME :- {Me_score} : COM :- {Com_score}")
# for volume inreasing function
def volumeup():
    for i in range(5):
        keyboard.press(Key.media_volume_up)
        keyboard.release(Key.media_volume_up)
        sleep(0.1)
# volume fecreasing function
def volumedown():
    for i in range(5):
        keyboard.press(Key.media_volume_down)
        keyboard.release(Key.media_volume_down)
        sleep(0.1)
# for searching something on google function
def searchGoogle(query):
        if "google" in query:
         import wikipedia as googleScrap
        query = query.replace("nova","")
        query = query.replace("google search","")
```

```
        query = query.replace("google","")
        speak("This is what i found on google")
        try:
            pywhatkit.search(query)
            result = googleScrap.summary(query,1)
            speak(result)
        except:
            speak("No speakable output available")
# searching something on you tube
def searchYoutube(query):
    if "youtube" in query:
        speak("This is what i found for your search!")
        query = query.replace("youtube search","")
        query = query.replace("youtube","")
        query = query.replace("nova","")
        web = "https://www.youtube.com/results?search_query=" + query
        webbrowser.open(web)
        pywhatkit.playonyt(query)
        speak("Done, Mam")
# searching something from wikipedia
def searchWikipedia(query):
    if "wikipedia" in query:
        speak("Searching from wikipedia....")
        query = query.replace("wikipedia","")
        query = query.replace("search wikipedia","")
        query = query.replace("nova","")
        Results = wikipedia.summary(query,sentences = 2)
        speak("According to wikipedia..")
        print(Results)
        speak(Results)
#for translating something
async def translategl(query):
    speak("Sure mam")
```

```
    print(googletrans.LANGUAGES)
    translator = Translator()
    # Await the translation
    text_to_translate = await translator.translate(query, src="auto", dest="hi")
    # Get the translated text
    text = text_to_translate.text
    print(f"Translated text: {text}")
    try:
        # Convert text to speech and save the file
        speakgl = gTTS(text=text, lang="hi", slow=False)
        speakgl.save("voice.mp3")
        playsound("voice.mp3")
        time.sleep(5)
        os.remove("voice.mp3")
    except Exception as e:
        print("Unable to translate")
        print(e)
# translating a text as a example function
def main():
    query = "translate"
    if "translate" in query:
        query = "hello"  # Example input to translate
        # Run the asynchronous translation function
        asyncio.run(translategl(query)
# after translation something  translation text with voice
def speak(text):
    """Function to speak a given text using gTTS."""
    tts = gTTS(text=text, lang='en', slow=False)
    tts.save("voice.mp3")
    playsound("voice.mp3")
    time.sleep(1)
    os.remove("voice.mp3")
# function for news in which u want
```

```
def latestnews():
    api_dict = {
        "business": "https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=464ecfbf36c349da888af8b33cb033f2",
        "entertainment": "https://newsapi.org/v2/top-headlines?country=us&category=entertainment&apiKey=464ecfbf36c349da888af8b33cb033f2",
        "health": "https://newsapi.org/v2/top-headlines?country=us&category=health&apiKey=464ecfbf36c349da888af8b33cb033f2",
        "science": "https://newsapi.org/v2/top-headlines?country=us&category=science&apiKey=464ecfbf36c349da888af8b33cb033f2",
        "sports": "https://newsapi.org/v2/top-headlines?country=us&category=sports&apiKey=464ecfbf36c349da888af8b33cb033f2",
        "technology": "https://newsapi.org/v2/top-headlines?country=us&category=technology&apiKey=464ecfbf36c349da888af8b33cb033f2"
    }
    recognizer = sr.Recognizer()
    mic = sr.Microphone()
    # Start listening for the field the user wants
    speak("Please tell me which field of news you want: business, entertainment, health, science, sports, or technology.")
    with mic as source:
        print("Listening...")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
    try:
        field = recognizer.recognize_google(audio).lower()
        print(f"Recognized command: {field}")
    except sr.UnknownValueError:
        speak("Sorry, I didn't catch that. Please try again.")
        return
    except sr.RequestError:
        speak("Sorry, I'm having trouble connecting to the service. Please try again later.")
```

```
    return
url = None
for key, value in api_dict.items():
    if key in field:
        url = value
        speak(f"Found news category: {key}")
        print(f"URL: {url}")
        break
if not url:
    speak("Sorry, I could not find the news category. Please try again.")
    return
response = requests.get(url)
news = response.json()
if news.get("status") == "ok":
    speak("Here is the first news.")
    arts = news["articles"]
    for articles in arts:
        article = articles["title"]
        speak(f"News headline: {article}")
        print(article)
        print(f"For more info, visit: {articles['url']}")
        # Ask whether to continue or stop based on voice command
        speak("Press one to continue or press two to stop.")
        with mic as source:
            print("Listening for continue or stop command...")
            recognizer.adjust_for_ambient_noise(source)
            audio = recognizer.listen(source)
        try:
            command = recognizer.recognize_google(audio).lower()
            print(f"Command recognized: {command}")
        except sr.UnknownValueError:
            speak("Sorry, I didn't catch that. Stopping the news feed.")
            break
```

```
            except sr.RequestError:
                speak("Sorry, I'm having trouble connecting to the service. Stopping the news feed.")
                break
            if command == "2":
                speak("Stopping the news feed.")
                break
            elif command != "1":
                speak("I didn't understand that. Stopping the news feed.")
                break
    else:
        speak(f"Error fetching news: {news.get('message')}")
# a list of  some apps which are in system
dictapp = {
    "commandprompt": "cmd",
    "paint": "mspaint",
    "word": "winword",
    "excel": "excel",
    "chrome": "chrome",
    "vscode": "code",
    "powerpoint": "powerpnt"
}
# temprory voice file making and after use it is remove it self
def speak(text):
    """Function to speak a given text using gTTS."""
    tts = gTTS(text=text, lang='en', slow=False)
    tts.save("voice.mp3")
    playsound("voice.mp3")
    time.sleep(1)
    os.remove("voice.mp3")
#for opening app function
def openappweb(query):
    speak("Launching, mam")
    # Check if the query contains a website URL
```

```
    if ".com" in query or ".co.in" in query or ".org" in query:
        query = query.replace("open", "")
        query = query.replace("kanha", "")
        query = query.replace("launch", "")
        query = query.replace(" ", "")
        webbrowser.open(f"https://www.{query}")
        speak(f"Opening {query}.")
    else:
        # Check if the query contains an app name from dictapp
        keys = list(dictapp.keys())
        for app in keys:
            if app in query:
                os.system(f"start {dictapp[app]}.exe")
                speak(f"{app} opened successfully.")
                return
        speak("App not found in the list.")
#for closing app function
def close_app(query):
    speak("Closing app, mam")
     # Full path to taskkill command
    taskkill_path = r"C:\Windows\System32\taskkill.exe"
    # Check if the query contains an app name from dictapp
    keys = list(dictapp.keys())
    for app in keys:
        if app in query:
            os.system(f"{taskkill_path} /f /im {dictapp[app]}.exe")
            speak(f"{app} closed successfully.")
            return
    speak("App not found to close.")
# for listening what we want to do close a app or open or exit
def listen_for_command():
    recognizer = sr.Recognizer()
    mic = sr.Microphone()
```

```
    speak("Please tell me the application or website you want to open or close.")
    with mic as source:
        print("Listening...")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
    try:
        command = recognizer.recognize_google(audio).lower()
        print(f"Command recognized: {command}")
        # Open or close app based on the command
        if "open" in command or "launch" in command:
            openappweb(command)
        elif "close" in command:
            close_app(command)
        else:
            speak("Sorry, I didn't recognize the command.")
    except sr.UnknownValueError:
        speak("Sorry, I didn't catch that. Please try again.")
    except sr.RequestError:
        speak("Sorry, I'm having trouble connecting to the service. Please try again later.")
# basic calculation function
def add(x, y):
      return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    else:
        return x / y
def calculator():
    print("Select operation:")
```

```
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")
    while True:
        choice = input("Enter choice(1/2/3/4): ")
        if choice in ('1', '2', '3', '4'):
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
            if choice == '1':
                print(f"{num1} + {num2} = {add(num1, num2)}")
            elif choice == '2':
                print(f"{num1} - {num2} = {subtract(num1, num2)}")
            elif choice == '3':
                print(f"{num1} * {num2} = {multiply(num1, num2)}")
            elif choice == '4':
                print(f"{num1} / {num2} = {divide(num1, num2)}")
        else:
            print("Invalid input")
        next_calculation = input("Do you want to perform another calculation? (yes/no): ")
        if next_calculation.lower() != 'yes':
            break
# function for clearing old tasks
def clear_old_tasks():
    speak("Do you want to clear old tasks? Please say YES or NO")
    query = takeCommand().lower()
    if "yes" in query:
        with open("tasks.txt", "w") as file:
            file.write("")
        return True
    elif "no" in query:
        return False
    else:
```

```
        speak("Invalid response. Assuming NO.")
        return False
#function for entering new tasks
def enter_tasks():
    tasks = []
    no_tasks = int(input("Enter the number of tasks: "))
    for i in range(no_tasks):
        task = input(f"Enter task {i+1}: ")
        tasks.append(task)
        with open("tasks.txt", "a") as file:
            file.write(f"{i+1}. {task}\n")
    return tasks
# function for scheduling day
def schedule_day():
    tasks = []
    clear_old = clear_old_tasks()
    no_tasks = int(input("Enter the number of tasks: "))
    for i in range(no_tasks):
        task = input(f"Enter task {i+1}: ")
        tasks.append(task)
        with open("tasks.txt", "a") as file:
            file.write(f"{i+1}. {task}\n")
# api key for chatbot like interaction
API_KEY="ddc-p7QmKYeLkG8FXMV8rSpedbSmhQ4xJ8fPWrL0KCig8d1ALrCTxW"
BASE_URL = "https://api.sree.shop/v1"
client = OpenAI(
    api_key=API_KEY,
    base_url=BASE_URL
)
# Initialize text-to-speech
engine = pyttsx3.init()
engine.setProperty('rate', 170)  # Set speaking rate
engine.setProperty('volume', 0.9)  # Set volume
```

```
# Speak function
def speak(text):
    """
    Convert text to speech.
    """
    engine.say(text)
    engine.runAndWait()
# Voice input function
def listen():
    """
    Listen to user input via microphone and convert it to text.
    """
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        recognizer.adjust_for_ambient_noise(source)  # Adjust for background noise
        try:
            audio = recognizer.listen(source, timeout=5, phrase_time_limit=10)
            print("Recognizing...")
            query = recognizer.recognize_google(audio, language='en-US')
            print(f"You said: {query}")
            return query.lower()
        except sr.UnknownValueError:
            speak("Sorry, I didn't catch that. Please try again.")
            return None
        except sr.RequestError:
            speak("There was an issue with the speech recognition service.")
            return None
# function for chat
def chat_completion(user_input):
    """
    Generate a normal chat completion response using the OpenAI API.
    """
```

```
    try:
        completion = client.chat.completions.create(
            model="gpt-4o",
            messages=[
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": user_input}
            ],
            temperature=0.7,
            max_tokens=1000
        )
        # Extracting the response content directly
        response_message = completion.choices[0].message.content
        return response_message
    except Exception as e:
        return f"An error occurred: {e}"

# function for entering in chatbot interactiona nd doing work
def meen():
    speak("Welcome! I am your voice assistant. How can I help you?")
    while True:
        speak("You can say 'Chat', or 'Exit' to interact with me.")
        query = listen()  # Listen for a voice command
        if query is None:
            continue  # Skip if no valid input is received
        elif "chat" in query:
            speak("What is your question?")
            user_input = listen()
            if user_input:
                response = chat_completion(user_input)
                print("\nChat Response:")
                print(response)
                speak("Here is my answer.")
                speak(response)
```

```
        elif "exit" in query or "quit" in query:
            speak("Goodbye! Have a great day!")
            break
        else:
            speak("I didn't understand that. Please try again.")
# function for showing schedule
def show_schedule():
    file = open("tasks.txt","r")
    content = file.read()
    file.close()
    mixer.init()
    mixer.music.load("notification.mp3")
    mixer.music.play()
    notification.notify(
        title = "My schedule :-",
        message = content,
        timeout = 15
    )
# function for sendng messages via whatsapp
def sendMessage():
    phone_number = "+91 8607565765 "#replace ph.no. as per demand
    message = "RADHE RADHE "#replace the message as per demand
    # Send message
    try:
        pywhatkit.sendwhatmsg(phone_number, message, time_hour=14, time_min=10)  # Adjust
time_hour and time_min as per your requirement
        print("Message sent successfully!")
    except Exception as e:
        print(f"An error occurred: {str(e)}")
  # start now functioning with commands.....
if __name__ == "__main__":
    while True:
        query = takeCommand().lower()
```

```
    if "radhe radhe" in query:
        greetMe()
        while True:
            query = takeCommand().lower()
            if "go to sleep" in query:
                speak("Ok , You can call me anytime")
                break
            elif "change password" in query:
                speak("What's the new password")
                new_pw = input("Enter the new password\n")
                new_password = open("password.txt","w")
                new_password.write(new_pw)
                new_password.close()
                speak("Done mam")
                speak(f"Your new password is{new_pw}")
            elif "professor details" in query:
                print("Hello, THIS project is assist by  ASSISTANT PROFFESSOR Mr. Anil Kumar ,Professor HEAD OF DEPATMENT DOCTOR BANTA SINGH JANGRA  . It's a pleasure to meet you.")
                print("Hello, Professor HEAD OF DEPATMENT DOCTOR BANTA SINGH JANGRA  . It's a pleasure to meet you.")
                speak("How may I assist you today? You can ask me questions, get information, or give me commands.")
                while True:
                    teacher_query = takeCommand()
                    if teacher_query == "":
                        continue
                    if teacher_query.lower() == "exit":
                        speak("Goodbye, Professor. Have a great day!")
                        break
                    elif "how are you" in teacher_query.lower():
                        speak("I'm just a program, Professor, but thank you for asking.")
                    elif "tell me a joke" in teacher_query.lower():
```

```
                speak("Why don't scientists trust atoms? Because they make up everything!")
            elif "what's the weather like" in teacher_query.lower():
                speak("I'm sorry, Professor, I cannot provide real-time weather information.")
            elif "set a reminder" in teacher_query.lower():
                speak("Sure, what would you like to be reminded of?")
                reminder = takeCommand()
                if reminder != "":
                    speak(f"Reminder set for {reminder}.")
                else:
                    speak("I'm sorry, Professor. I'm still learning and may not be able to assist with that yet.")
        elif "developer details" in query:
            speak("My developer is sonu. she created me to assist with various tasks and make life easier.")
            speak("If you have any questions or need further assistance, feel free to ask!")
        elif "schedule my day" in query:
         schedule_day()
        elif "show my schedule" in query:
         show_schedule()
        elif "play music" in query:
            play_music()
        elif"ask gpt" in query:
            meen()
        elif "translate" in query:
            main()
        elif "open file" in query:
            open_file()
        elif "open" in query:
            query = query.replace("open","")
            query = query.replace("kanha","")
            pyautogui.press("super")
            pyautogui.typewrite(query)
            pyautogui.press("enter")
```

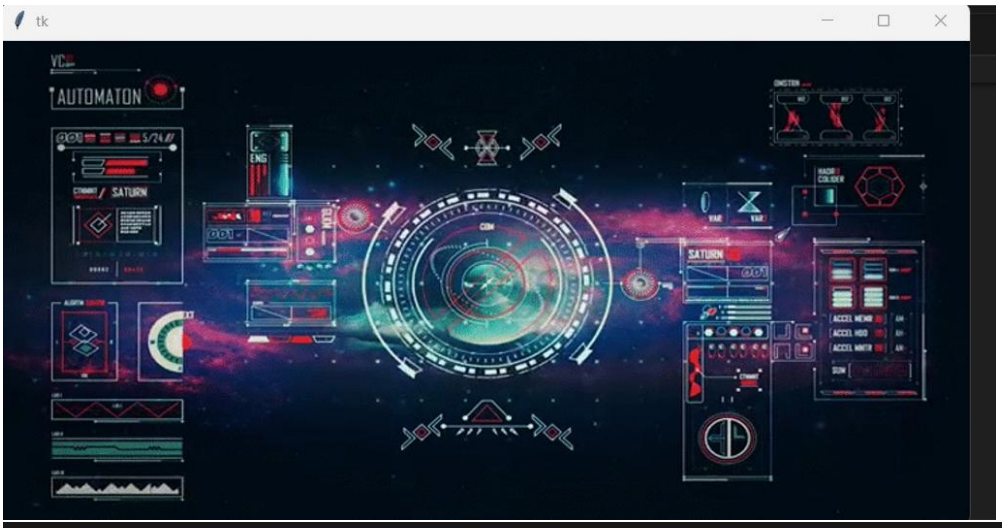
```
            elif "play a game" in query:
              game_play()
            elif "screenshot" in query:
                import pyautogui #pip install pyautogui
                im = pyautogui.screenshot()
                im.save("ss.jpg")
            elif "click my photo" in query:
                  pyautogui.press("super")  # Open start/search menu (Windows)
                  time.sleep(1)
                  pyautogui.typewrite("Camera")
                  time.sleep(1)
                  pyautogui.press("enter")  # Open Camera app
                  time.sleep(5)  # Wait for Camera app to open
                  pyautogui.press("enter")  # Take picture
                  speak("SMILE")
                  speak("Photo clicked successfully")
            elif "hello" in query:
                speak("Hello mam, how are you ?")
            elif "i am fine" in query:
                speak("that's great, mam")
            elif "how are you" in query:
                speak("Perfect, mam")
            elif "thank you" in query:
                speak("you are welcome, mam")
            elif "tired" in query:
             speak("Playing your favourite songs, ma'am")
              # List of URLs to choose from (Example: you can replace or add more URLs)
             urls = [
              "https://youtu.be/30YNd5fEGMo?si=Bbs3Bown5NSc","https://youtu.be/f18cpg-
mTrY?si=ZQFhG_EJuzque9eD","https://youtu.be/6ZwwapPikyQ?si=IySOi0gck0kGtwjP"

"https://youtu.be/Yppzo6dTpzY?si=6Uw_AZ4I9ZAJPuKG","https://youtu.be/lxKFe_1QVRA?s
i=n9YB6ej4TjhYLyrt"
```

```
    # Add more URLs here if needed
    ]
     # Choose a URL at random
 selected_url = random.choice(urls)
 webbrowser.open(selected_url)
elif "pause" in query:
    pyautogui.press("k")
    speak("video paused")
elif "play" in query:
    pyautogui.press("k")
    speak("video played")
elif "mute" in query:
    pyautogui.press("m")
    speak("video muted")
elif "volume up" in query:
    speak("Turning volume up")
    volumeup()
elif "volume down" in query:
    speak("Turning volume down")
    volumedown()
elif "app" in query:
    # from Dictapp import closeappweb
    listen_for_command()
elif "google" in query:
    searchGoogle(query)
elif "youtube" in query:
    searchYoutube(query)
elif "wikipedia" in query:
    searchWikipedia(query)
elif "news" in query:
    latestnews()
elif "calculate" in query:
    calculator()
```

```
elif "message" in query:
    sendMessage()
if "close file" in query or "close application" in query:
        close_file()
elif "temperature" in query:
    search = "temperature in hansi"
    url = f"https://www.google.com/search?q={search}"
    r  = requests.get(url)
    data = BeautifulSoup(r.text,"html.parser")
    temp = data.find("div", class_ = "BNeawe").text
    speak(f"current{search} is {temp}")
     elif "the time" in query:
    strTime = datetime.datetime.now().strftime("%H:%M")
    speak(f"mam, the time is {strTime}")
    print(strTime)
elif "finally sleep" in query:
    speak("Going to sleep,mam")
    exit()
elif "remember that" in query:
    rememberMessage = query.replace("remember that","")
    rememberMessage = query.replace("kanha","")
    speak("You told me to remember that"+rememberMessage)
    remember = open("Remember.txt","a")
    remember.write(rememberMessage)
    remember.close()
elif "what do you remember" in query:
    remember = open("Remember.txt","r")
    speak("You told me to remember that" + remember.read())
    print("You told me to remember that" + remember.read())
elif "search a song" in query:
    speak("Sure, what song would you like to search for on YouTube?")
    song_name = takeCommand()
    search_on_youtube(song_name)
```

```
elif "meeting" in query:
    speak("Ok mam opening meeet")
    webbrowser.open("https://meet.google.com/")
elif "check internet speed" in query:
    query = "check internet speed"
    check_internet_speed()
    speak("Here are the current internet speed metrics.")
    print("Here are the current internet speed metrics.")
elif "show my location" in query:
    current_location = get_current_location()
    print("Your current location is:", current_location)
    speak(f"Your current location is {current_location}.")
elif "generate qr code" in query:
    speak("Sure, please provide the text or URL for the QR code.")
    text_or_url = takeCommand().lower()
    text_or_url=input("enter the text")
    print(f"text or url is { text_or_url}")
    filename = "qr_code.png"
    generate_qr_code(text_or_url, filename)
    speak("QR code generated successfully.")
elif"shutdown system"in query:
    subprocess.call(['shutdown', '/s', '/t', '0'])
```

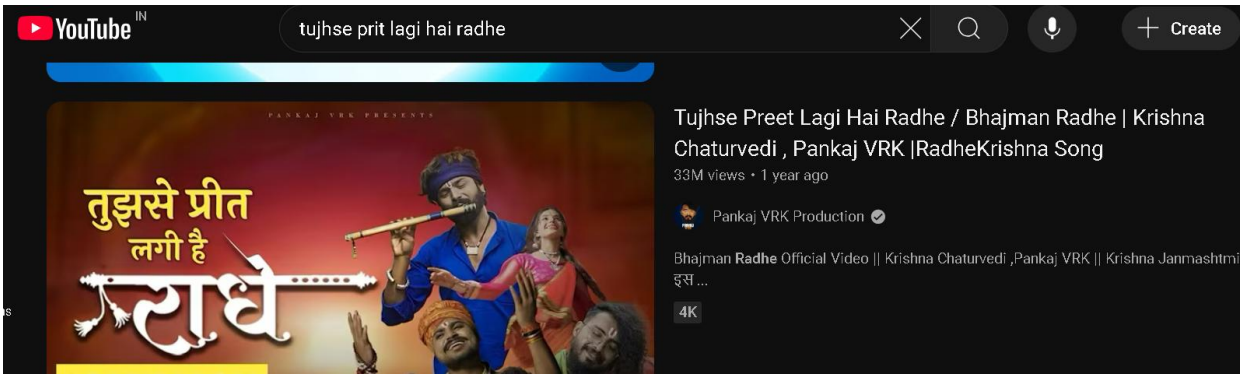
7.SCREENSHOTS



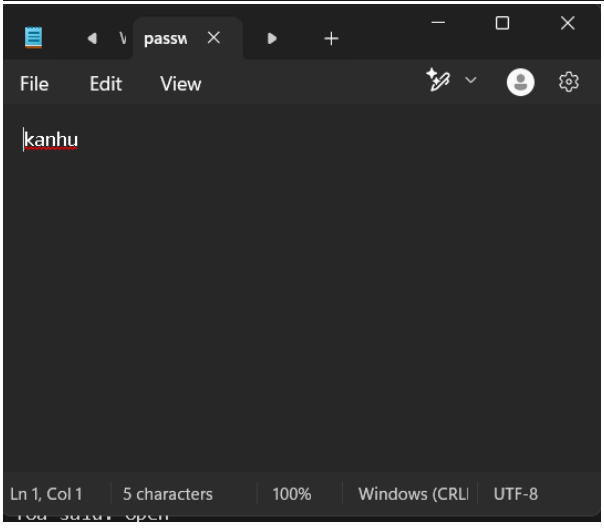

```
PS C:\Users\mehta\OneDrive\Desktop\projectfolder> & C:/Users/mehta/OneDrive/De
ctfolder/Jarvis_main.py
pygame 2.6.1 (SDL 2.28.4, Python 3.13.1)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter Password to get help from kanha :- kanhu
WELCOME ! PLZ SPEAK [radhe radhe] TO LOAD ME UP
Listening your command ...
Processing speech...
You said: Radhe Radhe
```

```
Listening your command ...
Processing speech...
You said: ask GPT

Listening...
Recognizing...
You said: chat chat chat chat chat chat chat chat chat
Listening...
Recognizing...
You said: what is cryptography

Chat Response:
Cryptography is the practice and study of techniques for securing
zed users. Its primary purpose is to ensure the confidentiality, i
```

```
Listening your command ...
Processing speech...
You said: search a song

Listening your command ...
Processing speech...
You said: Tujhse Prit Lagi Hai Radhe
```

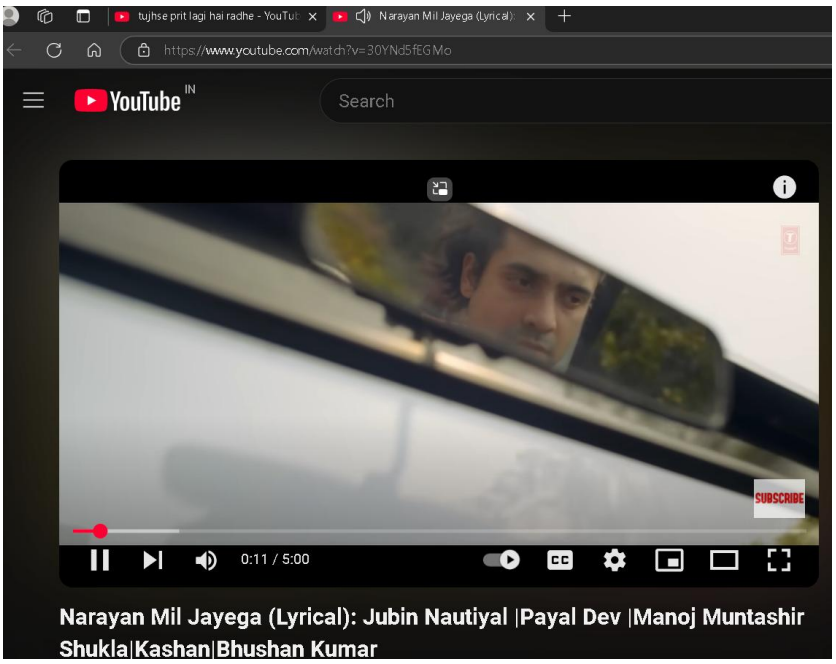



```
Listening your command ...
Processing speech...
You said: open file

Listening your command ...
Processing speech...
You said: password.txt
```



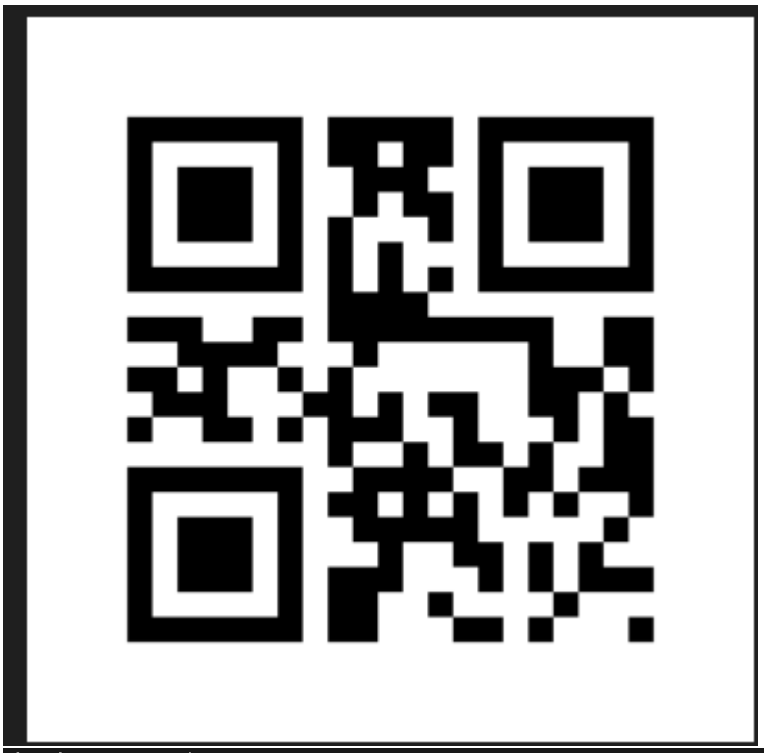
```
Listening your command ...
Processing speech...
You said: close file

Listening your command ...
Processing speech...
You said: password.txt
```

```
Listening your command ...
Processing speech...
You said: tired
```




```
Listening your command ...
Processing speech...
You said: news

Listening...
Listening your command ...
Processing speech...
You said: news

Listening...
Recognized command: sports
URL: https://newsapi.org/v2/top-headlines?country=us&category=sports&apiKey=464ecfbf36c349da888af8b33cb033f2
Oscar Piastri wins Chinese Grand Prix while Lewis Hamilton and two other drivers disqualified - CNN
For more info, visit: https://www.cnn.com/2025/03/23/sport/oscar-piastri-chinese-grand-prix-mclaren-spt-intl/index.html
```

```
Listening your command ...
Processing speech...
You said: generate QR code

Listening your command ...
Processing speech...
Couldn't understand the command. Please repeat.
enter the text1paytm.com
text or url is 1paytm.com
QR code generated successfully as qr_code.png
```



```
Listening your command ...
Processing speech...
Couldn't understand the command. Please repeat.
Listening your command ...
Processing speech...
You said: show my schedule

Listening your command ...
```

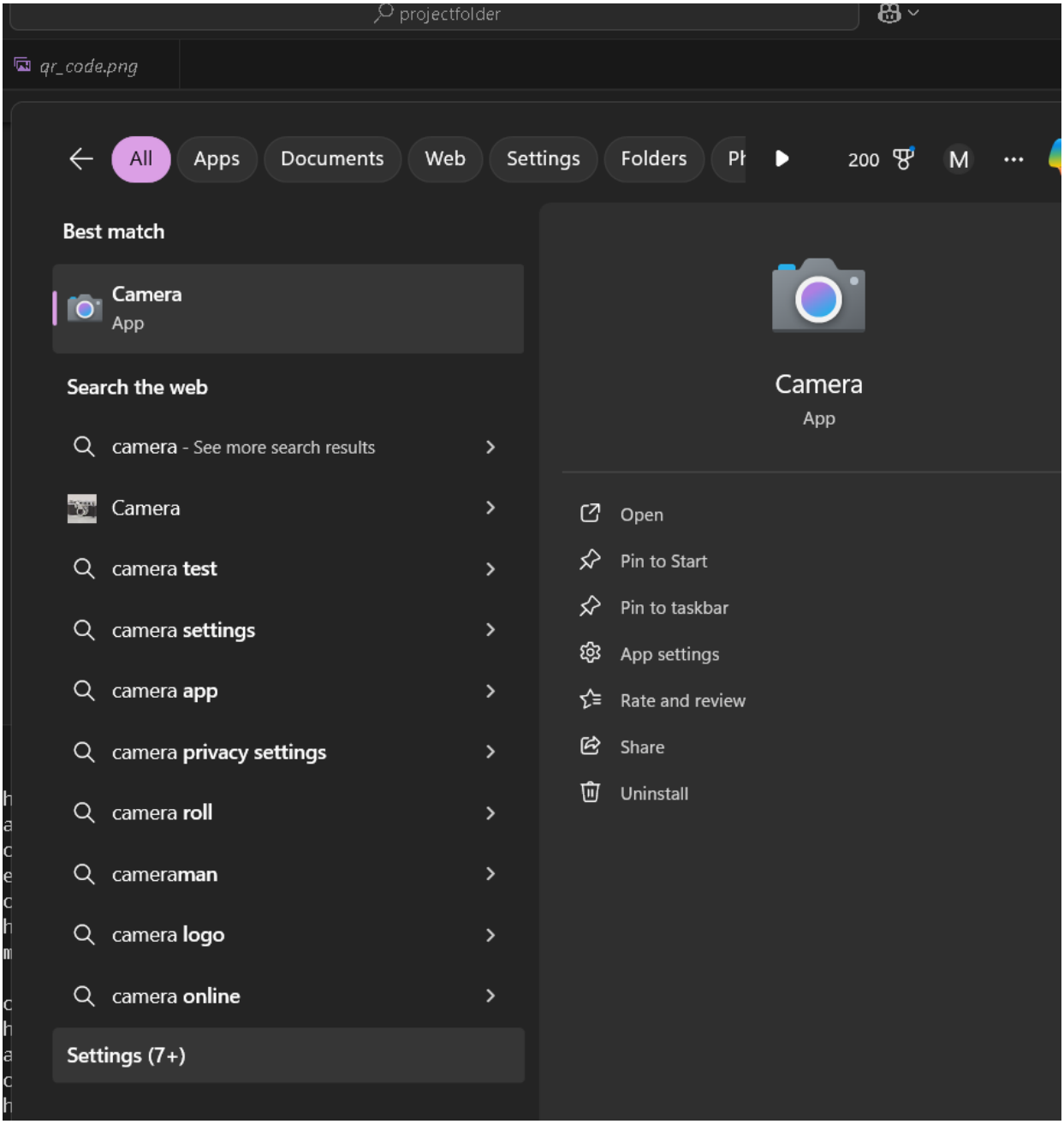
Python

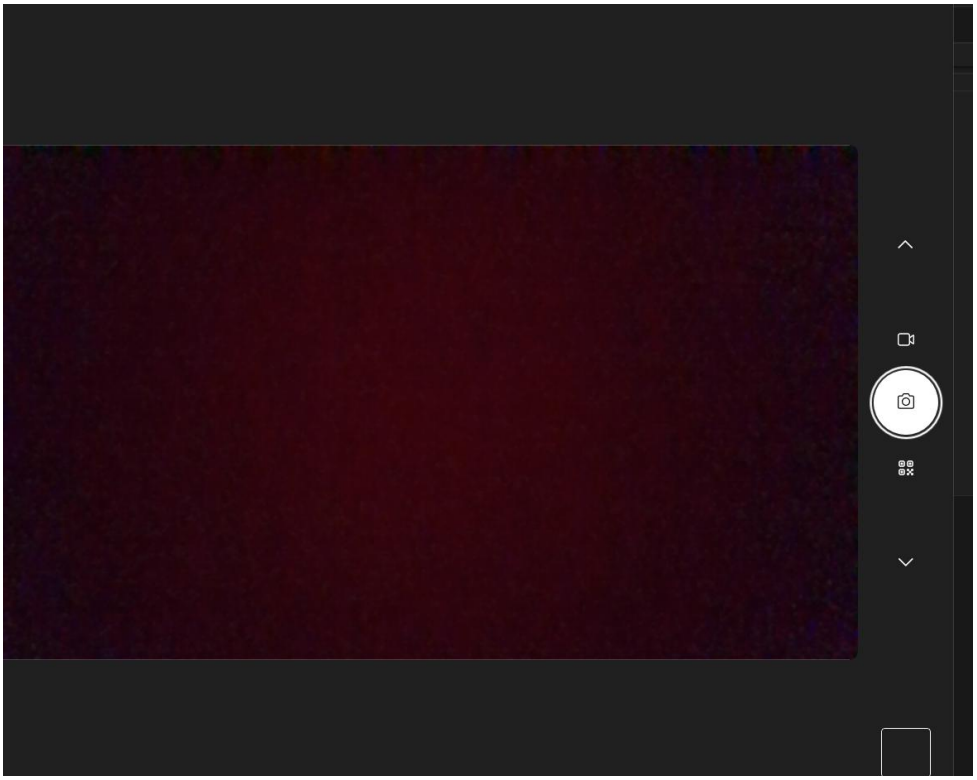
My schedule :-
0. coding at 1 pm
1. create video at 2 pm
2. upload code at 4 pm
1. do some rest at 8p.m.

```
Listening your command ...
Processing speech...
You said: volume down
```

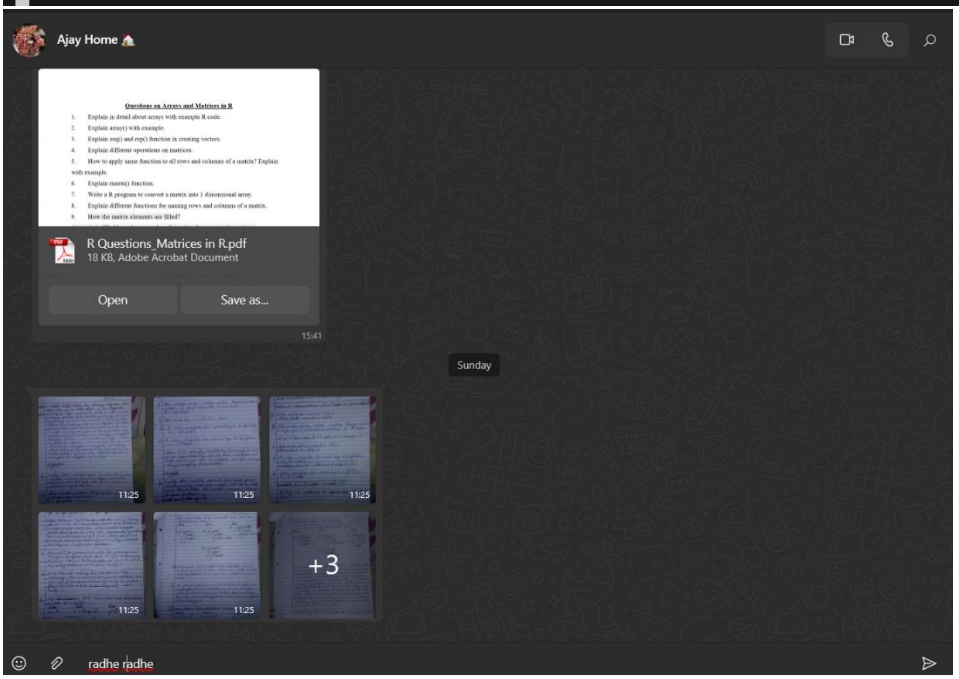
78

```
Listening your command ...
Processing speech...
You said: click my photo
```





```
Listening your command ...
Processing speech...
You said: message

In 8414 Seconds WhatsApp will open and after 15 Seconds Message will be Delivered!
```



7.Future Work

Enhanced Natural Language Understanding: Improve the assistant's ability to understand and process complex commands.

Expanded Language Support Integrate :additional languages and dialects for broader accessibility.

User Interface Improvements: Develop a more sophisticated graphical user interface for better user interaction.

Advanced Personalization: Implement machine learning to personalize responses and actions based on user preferences.

This report provides a comprehensive overview of the Virtual Assistant project, detailing its features, implementation, and future directions. The assistant exemplifies the application of AI and automation in everyday tasks, showcasing the potential for future advancements in virtual assistant technology.

8.Conclusion

The **Kanha Virtual Assistant** is a sophisticated and multifunctional AI-powered system designed to significantly enhance user productivity by simplifying everyday computing tasks. Leveraging **voice recognition, natural language processing (NLP), and multi-functional automation**, it offers a seamless and intuitive user experience. With its ability to interpret and execute a wide range of commands, the assistant serves as a dynamic digital companion capable of handling both simple and complex operations. From **web browsing, multimedia control, and task scheduling** to **system management and real-time information retrieval**, Kanha streamlines digital interactions with remarkable efficiency. Its conversational capabilities, coupled with **vocal responses and real-time feedback**, create a natural and engaging interface, making it easy to use even for non-technical users. Furthermore, with **error-handling mechanisms** in place, the assistant ensures smooth performance by gracefully managing invalid commands or failed operations. Designed for both personal and professional use, **Kanha Virtual Assistant** stands out as a reliable, intelligent, and user-centric solution, making daily computing tasks faster, smarter, and more convenient.